DESIGN

AUTOMATA.

:COMPUTATIONAL

MODELS

FRANK APPIAH

About Author:

Frank Appiah hold studentships at the Bsc. (Bachelor in computer engineering, *Kwame Nkrumah University*) student Level in 2009, Msc.(software engineering, *King's College London*) student Level in 2009/10 and 2 doctorates of philosophy in computer engineering from Kwame Nkrumah University of Science and Technology in 2013/14. Student Appiah created and developed a software platform called StreamEPS in the Java programming language. Appiah is a member of the National Academy of Engineering in Ghana.

Illustration Index

Illustration 1: Automatic Pen Controller.....21

Illustration 2: The Writing Automaton.....32

Illustration 3: The Key Automaton.....34

Illustration 4: Automatic Pen Controller.....49

Illustration 5: The Writing Automaton.....56

Illustration 6: Wusair.....87

Illustration 7: Wusair State Machine.....90

Index of Tables

Table1.....22

Table2.....32

Table3.....35

Table4.....50

Table5.....57

Table of Contents

About Author: 4

PREFACE.....14

DEDICATION.....15

I CHAPTER 1.....17

II True Depth of Von Neumann's Machine Revelation.....17

1 INTRODUCTION.....19

2 MACHINE REVELATION.....27

3 NEUMANN MACHINE COMPLEXITY...30

The State Transition Table.....35

4 HANDSIDE ARCHITECTURE.....37

SUMMARY.....41

REFERENCES.....43

III CHAPTER 2 45

IV A Computational Model of a Writing Automaton.....45

1 INTRODUCTION.....47

2 WRITING AUTOMATON.....54

The State Transition Table.....57

3 MACHINE PROCESSING OF WRITING FINITE AUTOMATON.....62

SUMMARY.....68

REFERENCES.....70

V CHAPTER 3.....72

VI WUSAIR AUTOMATON.....72

1 INTRODUCTION.....74

2 WUSAIR TASK ORIENTATION.....85

3 WUSAIR AUTOMATON.....88

4 STATE TRANSITION TABLE OF POUNDING AUTOMATON MODEL.....94

5 MACHINE PROCESSING OF WUSAIR AUTOMATON.....121

SUMMARY.....131

REFERENCES.....133

PREFACE

This teaches finite automaton in the design of a computing device explained with a pen or writing controller and fufu state machine. The true depth of Von Neumann's machine is revealed as well as the Von Neumann's complexity. The analogy of automatic pen to the self-reproduction is thoroughly described. A computational model of writing automaton and key automaton is stated with diagrams. A hand-side architecture is proposed as mathematical conditions for Von's machine. There is a further reading on a new computational transition function- A demonstration of computational science.

DEDICATION

THIS IS DEDICATED TO ALL TEACHERS IN SCHOOLS OR COLLEGE.

I CHAPTER 1

II True Depth of Von Neumann's Machine Revelation.

(Hand-side Architecture)

1 INTRODUCTION

A computational model[2] of finite automaton is introduced here to help in grasping the methodology used in this whole paper. Finite automata are good models for computers with an extremely limited amount memory. It is a manageable mathematical theory to an idealized computer with a small memory. I will describe a pen controller with finite automaton. The controller for a pen is a device for writing on paper. Often found is ink which is filled from the rear part and

normally writes from the front ball part. An automatic pen has a ball in front to script ink onto a paper when pressed in the presence of paper(white). In the automation, the ink can be exhausted and needs to be refilled without touching it physically. Devising this kind of pen machine is complex to some margin. I will look at this simply to the states that it can attain. The state transition diagram of the pen controller is shown below:

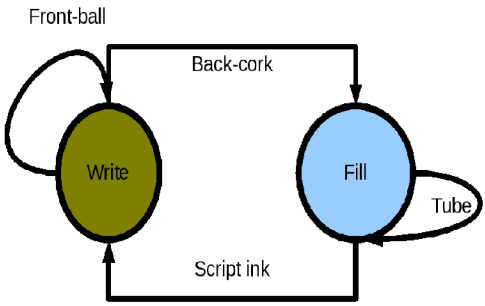


Illustration 1: Automatic Pen Controller

The controller is in only two states : “Write” or “Fill”, representing the corresponding condition of the automatic pen. As shown in the illustration 1, there are four possible input conditions: Front-ball(meaning that the pen writes several times in loop), Back-

cork(meaning the pen is out of ink and remove the back cork), Tube(meaning with opened back cork fill the tube with ink until full) and Script-ink(closed cork with the script ink and start writing again). The next thing to do is the state transition table.

State Transition Table of Automatic Pen

State	**Input Signals**			
	Front-ball	**Back-cork**	**Tube**	**Script- ink**
Write	Write	Fill	Fill	Write

Fill	Write	Fill	Fill	Write

The controller moves from state to state, depending on the input it receives.

1. In the state *Write*, the controller receiving a *front-ball* signal will continue to write until the ink is finish.
2. In the state *Write*, the controller receiving a *back-cork* signal will open the back cork of the pen and to fill it with ink.
3. In the state *Write*, the controller receiving a *tube* signal will fill the tube with ink.

4. In the state *Write*, the controller receiving a *script-ink* signal will stop the filling and start to write again. **AND**
5. In the state *Fill*, the controller receiving a *front-ball* signal will start to write until the ink is finish.
6. In the state *Fill*, the controller receiving a *back-cork* signal will open the back cork of the pen and to fill it with ink again.
7. In the state *Fill*, the controller receiving a *tube* signal will continue filling the tube with ink.

8. In the state *Fill*, the controller receiving a *script-ink* signal will stop the filling and use the *script-ink* to write again.

In section 2, the real depth of Von Neumann machine is revealed , then to Neumann Machine Complexity in section 3 that describes the analogy of self-reproducing machine and pen. The writing automaton and key automaton are described as state diagrams and the state transitions of the automaton are tabulated. In section 4, the Hand-side Architecture is put forward and described as a two mathematical conditions for insuring self-reproduction. A hand sketch is

illustrated along with the hand-side mappings of the computer. The details of this paper titled as Hand-side Architecture is idealized without any questions.

2 MACHINE REVELATION

I claim to reveal the true depth of Von Neumann machine on the hand-side of its architecture . The hand-side of the architecture is an attempt to resolve the use of the hand part of the human body as an issue first then the use of memory. This influences the development of the Von Neumann's machine. The hand presses on the keyboard (fingers touch) to type information less human memory in the universal constructor[1, 3]. The

new antecedent generates a whole family of new consequent problem. This results to the obvious substantial to express its real force to formulate such way that cannot be trivialized. The true depth of having a lasting informed memory is only by hand writing. The units of the current computer[4] based on Non Neumann's machine are memory unit, ALU unit, processor unit, input unit and output unit. The processor unit is made up data-path and control components. The hand has five (5) parts of fingers and the coincidence of this is the realized five units of the current computer . The real force of the hand helped

to architect the Von Neumann 's machine . The pen is a machine held in the hand that writes word(s) on output paper. The self-reproduction[3] of an earlier recorded information on paper has machined.

3 NEUMANN MACHINE COMPLEXITY

The complexity of Von Neumann 's machine is the number of units employed in the self-reproduction. The self-reproducing automaton requires a pen as a keyboard which will write as to type information onto paper as onto memory unit and will be recorded on paper as to output unit for display. Von Neumann 's design is small and complex ,and relies on the

exact operation and intricate interactions between the many simple parts. The pen is a simpler machinery than the computer from Turing.

The Writing Automaton

The pen inks as it writes words as information (info) onto a paper and the paper records to output as a note.

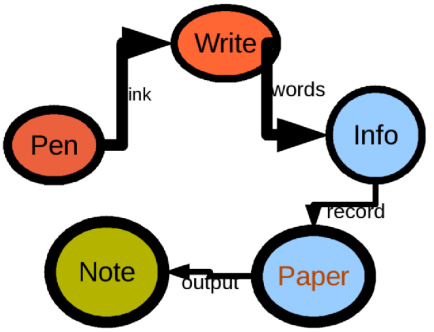


Illustration 2: The Writing Automaton

The State Transition Table

States	Ink	Words	Record	Output
Pen	Write	Info	Paper	Note
Write	Empty	Info	Paper	Note

Info	Empty	Empty	Paper	Note
Paper	Empty	Empty	Empty	Note
Note	Empty	Empty	Empty	Accept

The Key Automaton

The key types information (info) as words to the memory as the recorded words are to the monitor for output to screen.

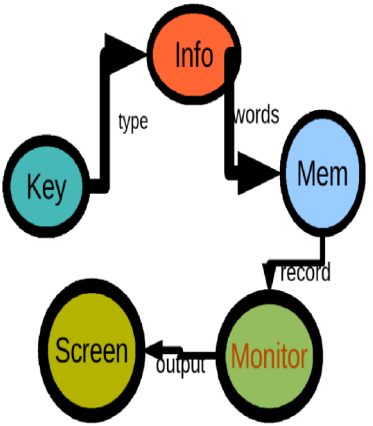


Illustration 3: The Key Automaton

The State Transition Table

States	**Type**	**Words**	**Record**	**Output**
Key	Info	Mem	Monitor	Screen
Info	Empty	Mem	Monitor	Screen
Mem	Empty	Empty	Monitor	Screen
Monitor	Empty	Empty	Empty	Screen
Screen	Empty	Empty	Empty	Accept

Von Neumann on himself does

(1) always eminently to construct a simple machine but in self-producing this simple ma-

chine express a sense of complexity in the self-reproduction.

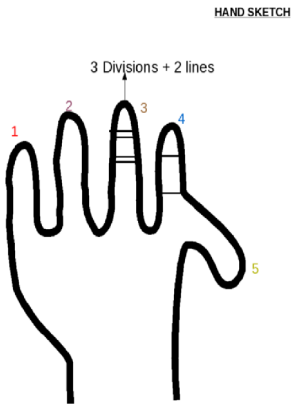
(2) is a machine -reproduction of (1). The complexity of Von Neumann's is the number of components of the computer. The complexity was described by Von Neumann as “vague , unscientific and imperfect” [1]. The existence of a self-reproducing universal hand -constructor in itself is relevant to machine self-reproduction.

4 HANDSIDE ARCHITECTURE

The mathematical conditions to insure self-reproduction are :

(i) The *five* finger parts of a hand and

(ii)The *two* part decomposition of hand components (left hand /right hand).



Hand-side Mappings

1: Memory Unit

2: ALU Unit

3: Input Unit

4: Output Unit

5: Processor Unit.

The machine self-reproduction involves the hand with five fingers and in its two parts (left

and right hands). The machine is self-reproduced because it assumes the mathematical conditions. The classic parts of the computer is five and it can be assumed from the mathematical condition (1). The two part decomposition of the processor as data-path and control is assumed on the mathematical condition(2). The right part of the processor is the control component and the left of the processor is the data-path component. It is satisfactory to reveal that Von Neumann machine does work on these mathematical conditions. It is a point view by me on the machine self-reproduction.

Assumptions

(i)The mathematical condition (1) assumes that the classic computer is made up of five units just as the 5 fingers of the hand.

(ii)The mathematical con-dition (2) assumes that the two parts of the processor is decomposed from the two-part decomposition of the hand (left/right).

SUMMARY

A process is mathematical in nature. Therefore the self-reproduction is a process. The hand-side architecture is a 25-state, two dimensional realized configuration embedded in a self-reproducing automata.

The notion of complexity is no longer vague because we know clearly the number of units based on the bioinspired hand-side architecture.

The notion of complexity is described as scientific based on the bioinspiration that caused the deduction of mathematical condition. The phrase "A pen held in hand that writes is perfect " is an expression of perfection on the logical perspective. In the machine self-reproduction sense, you can put it as " A key typed to put information as a record in memory that output as a display is perfect".

REFERENCES

1. *Barry McMullins. John von Neumann and the Evolutionary Growth of Complexity: Looking Backwards, Looking Forwards, MIT Press (2000).*
2. *Micheal Sipser. Introduction to the Theory of Computation, PWS Publishing Company, (1997).*
3. *Arthur W. Burks, editor. Theory of Self-Reproducing Automata, University of Illinois Press, Urbana, (1966).*
4. *Yoshihide Igarashi, Tom Altman, Mariko Funada and Barbara Kamiyama. Computing: A Historical and Technical Perspective. CRC Press, Taylor &Francis Group (2014).*

III CHAPTER 2

IV A Computational Model of a Writing Automaton.

(Automatic Pen Controller)

1 INTRODUCTION

A computational model[2] of finite automaton is introduced here to help in grasping the methodology used in this whole paper. Finite automata are good models for computers with an extremely limited amount memory. It is a manageable mathematical theory to an idealized computer with a small memory. I will describe a pen controller with finite automaton. The controller for a pen is a device for writing on paper. Often found is ink which is filled from the rear part and normally

writes from the front ball part. An automatic pen has a ball in front to script ink onto a paper when pressed in the presence of paper(white). In the automation, the ink can be exhausted and needs to be refilled without touching it physically. Devising this kind of pen machine is complex to some margin. I will look at this simply to the states that it can attain. The state transition diagram of the pen controller is shown below:

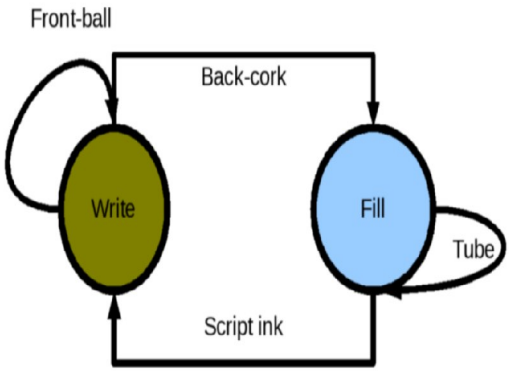


Illustration 4: Automatic Pen Controller

The controller is in only two states : "Write" or "Fill", representing the corresponding condition of the automatic pen. As shown in the illustration 1, there are four possible input conditions: Front-ball(meaning that the pen writes several times in loop), Back-

cork(meaning the pen is out of ink and remove the back cork), Tube(meaning with opened back cork fill the tube with ink until full) and Script-ink(closed cork with the script ink and start writing again).

The next thing to do is the state transition table.

State Transition Table of Automatic Pen

States	**Input Signals**			
	Front-ball	**Back-cork**	**Tube**	**Script-ink**

Write	Write	Fill	Fill	Write
Fill	Write	Fill	Fill	Write

The controller moves from state to state, depending on the input it receives.

1. In the state *Write*, the controller receiving a *front-ball* signal will continue to write until the ink is finish.
2. In the state *Write*, the controller receiving a *back-cork* signal will open the back cork of the pen and to fill it with ink.

3. In the state *Write*, the controller receiving a *tube* signal will fill the tube with ink.
4. In the state *Write*, the controller receiving a *script-ink* signal will stop the filling and start to write again. **AND**
5. In the state *Fill*, the controller receiving a *front-ball* signal will start to write until the ink is finish.
6. In the state *Fill*, the controller receiving a *back-cork* signal will open the back cork of the pen and to fill it with ink again.
7. In the state *Fill*, the controller receiving a *tube* signal will continue filling the tube with ink.

8. In the state *Fill*, the controller receiving a *script-ink* signal will stop the filling and use the *script-ink* to write again.

2 WRITING AUTOMATON

The writing automaton requires a pen as a keyboard which will write as to type information onto paper as onto memory unit and will be recorded on paper as to output unit for display. The pen is a simpler machinery than the computer from Turing.

The Writing Automaton

The pen inks as it writes words as information (info) onto a paper

and the paper records to output as a note.

The writing controller is in only five states : “Pen”, “Info”, “Paper”, “Note” or “Write”, representing the corresponding conditions of the writing controller. As shown in the illustration 2, there are four possible input conditions: Ink(meaning that the pen writes with ink), Write(meaning the pen writes words in ink), Info(meaning words are presented as information) , Paper (meaning information is recorded on paper) and Note(meaning the

information recorded on paper is outputted as a note).

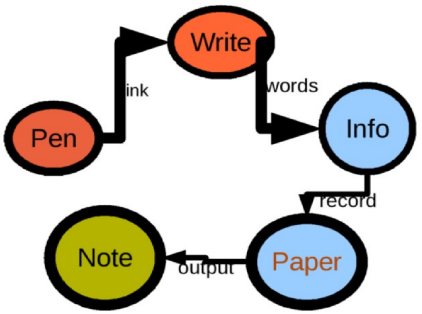


Illustration 5: The Writing Automaton

The State Transition Table

States	**Input Signals**			
	Ink	**Words**	**Record**	**Output**
Pen	Write	Info	Paper	Note
Write	Empty	Info	Paper	Note
Info	Empty	Empty	Paper	Note
Paper	Empty	Empty	Empty	Note
Note	Empty	Empty	Empty	Accept

The writing controller moves from state to state, depending on the input it receives.

1. In the state *Pen*, the controller receiving an *ink* signal will start to write.

2. In the state *Pen*, the controller receiving *words* signal will move to the info state.

3. In the state *Pen*, the controller receiving *record* signal will move to the *paper* state.

4. In the state *Pen*, the controller receiving *output* signal will note as it moves to that state. ***AND***

5. In the state *Write*, the controller receiving an *ink* signal will not move to any state, empty.

6. In the state *Write*, the controller receiving *words* signal will move to the info state.

7. In the state *Write*, the controller receiving *record* signal will move to the *paper* state.

8. In the state *Write*, the controller receiving *output* signal will *note* as it moves to that state. ***AND***

9. In the state *Info*, the controller receiving an *ink* signal will not start to write.

10. In the state *Info*, the controller receiving *words* signal will not move to the info state.

11. In the state *Info*, the controller receiving *record* signal will move to the *paper* state.

12. In the state *Info*, the controller receiving output signal will *note* as it moves to that state. ***AND***

13. In the state *Paper*, the controller receiving an *ink* signal will not start to write.

14. In the state *Paper*, the controller receiving *words* signal will not move to the info state.

15. In the state *Paper*, the controller receiving *record* signal will not move to the *paper* state.

16. In the state *Paper*, the controller receiving *output* signal will *note* as it moves to that state**.** ***AND***

17. In the state *Note*, the controller receiving an *ink* signal will not move to write.

18. In the state *Note*, the controller receiving *words* signal will not move to the *info* state.

19. In the state *Note*, the controller receiving *record* signal will not move to the *paper* state.

20. In the state *Note*, the controller receiving output signal will note as it moves to that state.

3 MACHINE PROCESSING OF WRITING FINITE AUTOMATON

Illustration 2 is a state transition diagram of the Writing Finite Automaton(***WFA***) called M_1. It has five states labeled *Pen, Info, Write , Paper and Note* ,and four conditions labeled *ink, record, words and output*. The start state of WFA is pen and it is normally indicated by a

pointing arrow to the state Pen. The accept state is the *note* state and it is normally indicated by double circle around the state (omitted here). The arrows moving from one state to another is called transitions. When the automaton receives an input string {pen, write, info, paper, note}, it processes that string and produces an output. The output is either accept or reject. The processing begins in M_1 start state. The automaton receives the symbols from the input string one by one from left to right. After reading the symbols, M_1 moves from one state to another along the transition that has symbol as its label. When it

reads the last symbol now it is in the accept state. The processing of M_1 as follows:

1. start in state *pen*;
2. read *pen*, follow transition from *pen* to *write*;
3. read *write*, follow transition from *write* to *info*;
4. read *info*, follow transition from *info* to *paper*;
5. read *paper*, follow transition from *paper* to *note*;
6. accept because M_1 is in an accept state *note* at the end of the input.

Transition function of Writing Finite Automaton

The transition function is used to define the rules of moving. The notation of the transition function is δ(state, input)=state.

```
1. δ(pen, ink) = write;
2. δ(pen, words) = info;
3. δ(pen, record) = paper;
4. δ(pen, output) = note;
   ------------------------
5. δ(write, ink) = empty;
6. δ(write, words) = info;
7. δ(write,record)= paper;
8. δ(write,output) = note;
   ------------------------
```

9. δ(info, ink) = empty;
10. δ(info, words) = empty;
11. δ(info,record) = paper;
12. δ(info, output) = note;

---

13. δ(paper, ink) = empty;
14. δ(paper,words) = empty;
15. δ(paper,record)= empty;
16. δ(paper,output)= note;

---

17. δ(note, ink) = empty;
18. δ(note, words) = empty;
19. δ(note,record) = empty;
20. δ(note,output)= **accept.**

We now arrived at the formal definition of finite automaton.

*A **finite automaton** is a 5-tuple (Q, Σ , δ, q , F), where*

1. Q *is a finite set called the states,*
2. Σ *Is a finite set called the alphabet,*
3. δ: $Q \times \Sigma \rightarrow Q$ *is the transition function,*
4. $q_0 \in Q$ *is the start state.*
5. $F \subseteq Q$ *is the set of accept states.*

A Writing Finite Automata is defined as :

1. Q: {*pen*, *write*, *info, paper, note*}

2. Σ: {*ink, words, record, output*}

3. δ: {*check transition functions above*}

4. qo is *pen*

5. F: {n*ote, output*}.

SUMMARY

This teaches the machine processing of writing finite automaton with a brief description of of an automatic pen controller. This controller functions on the carry out a transition function that takes two states and outputs a new resulting state. There are about 20 transition functions in the design automaton. The computing device is mechanized as the writing automaton computes. The computational model of the writing automaton is based on the states of the writing automaton diagram.

REFERENCES

1. *Barry McMullins. John von Neumann and the Evolutionary Growth of Complexity: Looking Backwards, Looking Forwards, MIT Press (2000).*
2. *Micheal Sipser. Introduction to the Theory of Computation, PWS Publishing Company, (1997).*
3. *Arthur W. Burks, editor. Theory of Self-Reproducing Automata, University of Illinois Press, Urbana, (1966).*
4. *Yoshihide Igarashi, Tom Altman, Mariko Funada and Barbara Kamiyama. Computing: A Historical and Technical Perspective. CRC Press, Taylor &Francis Group (2014).*

v CHAPTER 3

VI WUSAIR AUTOMATON.

A Pounding Task Design.

1 INTRODUCTION

The simulation of pistol-mortar activity is a real-world systems with a level of simple details. The interaction of subsystems includes the pistol, material(cassava) and mortar. The pounding skill is an inputing changes to gain skill in designing a machine to do as it sounds. This simulation study of the traditional pistol-mortar pounding task is described thoroughly in the description section to

mimic the real-world activity of fufu pounding. A pistol-mortar assembly (*pma*) is a line of worker in the non-industrial era or domestic homes but still in practice. A million *pma* is engaged weekly at various homes and that is a source of technological change[4]. A pma machine is a diminishing of workload at domestic homes. The research draws it findings on the Wusair Task Graph (WTG). This graph describes the observation of the pounders in the game of pounding. It is inspired by biological observation of breathing by both

pounders. Based on the activity of pistol-mortar of Ashanti, a pounding task is created and that is called task-oriented design. The wusair graph of task-orientation of pistol-mortar activity is shown. The simulations of the graph are observed in this teaching. Based on the activity of pistol-mortar of Ashanti, a pounding task is created and that is called task-oriented design. The wusair graph of task-orientation of pistol-mortar activity is shown. The simulations of the graph are observed in this teaching. The purpose is to describe a

simulation of what pistol-mortar activity has to do exactly what it sounds like they should do in real-world [1].The pound-player is interested in re-creating real-world conditions on some limited basis for the purpose of aping or mimicking the behavior of the real-world system. A primate evolution resulted in the history of work done mentally and physically. The facts and concepts developed are our work and technology. A simulation is a model that copies the behavior of some aspect or aspects of reality[3, pg 187]. Certain

characteristics of simulations make them specific to a form[3, pg 187]. Simulations deal with real-world systems with a level of detail and are interested in the interactions of subsystems that exist in the overall system that is being studied. Simulations are system specific, dealing with individual problems or scenarios to allow the participant to gain experience artificially in interacting with real-world problems and also are both quantitative and qualitative, offering probable outcomes to

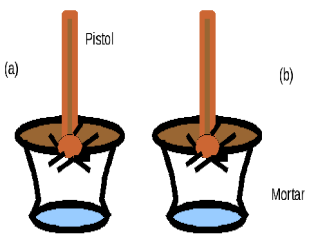
given actions. Finally, simulations are designed to mimic real-world behavior.

Lets look at the simulation of a pistol-mortar graph activity.

DESCRIPTION

A typical home of an Ashanti has many pounding task. A pounding task involves the preparation of fufu for a family at Ashanti home. Fufu is a popular disk for such homes in Ashanti Region. It is normally prepared as a super delicacy for

evenings. It is a large menu of some kind and it is prepared from cassava, plantain or cocoyam. A preparation from cassava and plantain is yellowish in color and that made from cassava and cocoyam is brownish in color.



Drawing 1:

Complete Pistol and Mortar from [2]

A POUNDING AUTOMATON MODEL

The figure above shows the two main parts of the pounding fufu design – A pistol and mortar. It is normally a brown-wooden pistol and

brown-wooden mortar[2]. A pistol is normally rounded from the top-to-bottom with spreading base. The spreading base is made by hitting the original stick on a concrete mortar (cement). The pressure of hitting and continues circular turning gives it a perfect spreading base. The strength limitation can halt the pounding for a short while and pounding will continue afterwards. Sometimes the hands are shuffle from left-to-right or right-to-left to ease pain in the palms of the hand. A pain relief method in the pounding process. This is a

pounding machine. A typical home, pounding is done by the boys and mothers. The hunting agent (hunter) normally brings the a home bush meat as we call it but other meats like goat, sheep, chicken, poultry, etc. The smoke fishes are a favorite source of protein for their soups- Our Ashanti Soups. It is classic at Ashanti Region and parts of Ghana.

Lets leave the fufu pounding task and name the pounding task itself as a "*Wusair*" - A task-oriented design. Wusair is pronounced as wasair. This name is chosen because the pounding task is stressful and consumes a lot of air in the metabolic process of the human body. More energy is needed to go through the process of pounding of fufu. A hard breathing process of fufu disk. Thanks to the famous

mechanical engineer and it is a pass thing now but still some homes do pound – *A pound-for-pound*. The pounding task involves a lot of looking down into the mortar unless the pistol will hit the stirrer sitting around the mortar. It is termed as an exercise for the young and married couples. Most married couples do not have children in the early ages of marriage and may be involved in *couple-pounding*. Couple pounding involves the process of performing a pounding task with a wife and a husband. A *self-pounding* case is also done in the home if the children of the home are within the ages of 1-10 and are not strong enough to perform the task. Self-pounding is the process of performing a pounding task without a pounder. In self-pounding, the pounder and stirrer are the same person. It is done by sitting and hitting the pistol in the mortar at a less pace

than the normal activity. There is a normal neck issue in the process of pounding and needs a moment stop for resting on the tired neck. It is a natural reasoning mechanism in the processing of fufu. A tired neck cannot occur without a tired feet or a tired neck resting on the shoulders of a stressful head.

2 WUSAIR TASK ORIENTATION

It is time to look at the task orientation of the pounding task called Wusair. The task orientation is based on the case when two pounders (a or b) and stirrers (a or b) are in a game of pounding and their air breathing can cause an observation. In order for an exercise is to be a game [1].These observations are described as following:

1. There is a pounder and a stirrer for each game.
2. Game, *a* pounder pounds at a pace as game, *b* pounder.
3. This is the driving task at hand.
4. Game, *a* and game, *b* will pound at the same for sometime and make a change. Game changes include one-hand pounding or double hand pounding or drinking water, cleaning sweat, easing pain in palm or taking in air or attending to soup or bowl placing for sharing.
5. Game, *a* pounder can rest for a moment whiles *b* continues pounding.

6. Game *b* will definitely come to a rest after sometime after *a*.
7. *a* or *b* will come back to the pounding task after a short rest.
8. Tasks 4-7 will continue until the pounding task is completed by the *a's* or *b's* task driver or driving task.

Wusair Task Graph

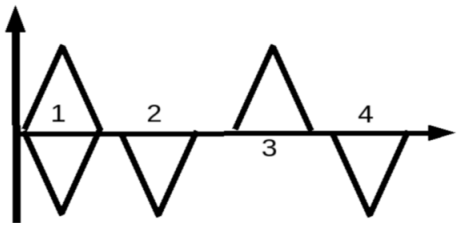


Illustration 6: Wusair

3 WUSAIR AUTOMATON

It requires a *boy*/*man* who will handle the *pistol* by moving it up and down. By so doing hitting it into the *mortar* and pounding the picks of *cassava* or *cocoyam*/*plantain* by the *girl*/*woman* sitting around the mortar – There by stirring and continuously adding little *water* to the pound material for it to soften/smoothen. There are two main mix in the preparation of fufu namely *cassava mix* and *cocoyam*/*plantain mix*. The pounding of picks of cassava will yield the *cassava mix* and

that of the cocoyam/plantain picks will yield the *cocoyam/plantain mix*. There is a recombination of both mixes in the mortar again for a soft and smooth fufu. Finally the state of fufu ends and it is a *dish* for eating.

Wusair Automaton

The pistol *hits to pound the* cassava *or* cocoyam/plantain *materials in a* mortar *separately but will finally recombine with little* water *added or sprinkled on the recombinant mixes to yield* fufu *state and ends in a* dish *bowl for eating.*

State Transition Diagram

Below is the state transition diagram depicting the fufu pounding state-machine.

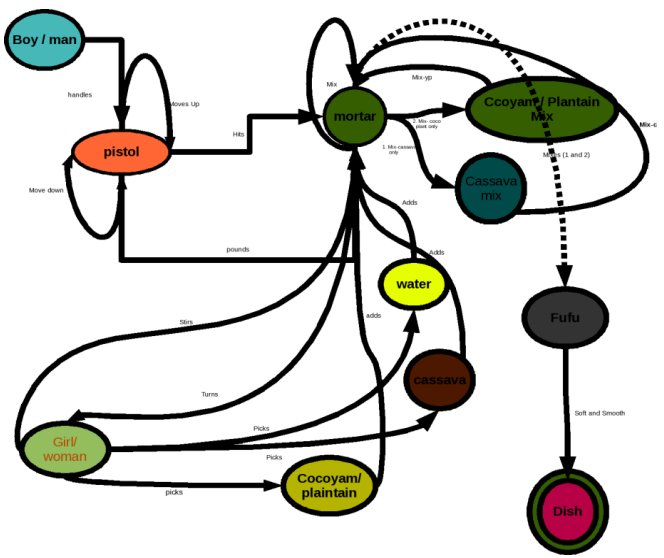


Illustration 7: Wusair State Machine

The pounding controller has 11 states:

> "boy/man", "pistol", "girl/woman", " mortar", "fufu", "cassava", "cassava mix","cocoyam / plantain", "cocoyam/ plantain mix", "water" and "dish" – representing the corresponding conditions of the pounding controller.

There are 20 possible input conditions: *handle* (meaning a boy/man handles a pistol), *move-up* (meaning a pistol moves up in handling it), *move-down*(meaning a pistol moves down in the handling process), *hits* (meaning a pistol moves up and down to hit a mortar), *pounds* (meaning material will be pounded in the hitting process in a pistol-mortar activity),

stirs (meaning a girl or woman sits around a mortar and stir the material together or separately), *turns* (meaning a girl or woman sits around a mortar and turns the materials to smooth them together), *picks* (meaning a woman or girl picks water to soften the materials being mixed or picks a cassava to pound or picks a cocoyam/plantain to pound), *adds* (meaning a woman or girl adds water to soften the materials being mixed or adds a cassava to pound or adds a cocoyam/plantain to pound), *mix-cassava-only* (meaning cassava added for pounding mixes with itself only after a number of picks), *mix-cocoplant-only* (

meaning cocoyam or plantain added for pounding mixes with itself only after a number of picks), *mix-yp* (meaning output of mix-cocoplant-only state is further sent to the mortar for mixing again), *mix-c* (meaning output of mix-cassava-only state is further sent to the mortar for mixing again), mix (meaning a loop to continuously mix all mixes until soften and smoothen), *mix-1-2* (meaning mix and pound both mixes in conditions mix-cocoplant-only and mix-cassava-only and soften-smoothen (meaning fufu state will be dished out when mixes are soften and smoothen).

4 STATE TRANSITION TABLE OF POUNDING AUTOMATON MODEL

The state transition table can be found at the appendix section.

The pounding controller moves from state to state depending on the input signal it receives:

1. In a **b/m** state, the controller receiving **H** signal will move to a **P** state.

2. In a **b/m** state, the controller receiving **Mu** signal will move to a **P** state.

3. In a **b/m** state, the controller receiving **Md** signal will move to a **P** state.

4. In a **b/m** state, the controller receiving **Hi** signal will move to a $\mathbf{M^2}$ state.

5. In a **b/m** state, the controller receiving **Po** signal will move to a **P** state.

6. In a **b/m** state, the controller receiving $\mathbf{Tu^2}$ signal will move to a **g/w** state.

7. In a **b/m** state, the controller receiving **S** signal will move to a **M** state.

8. In a **b/m** state, the controller receiving $\mathbf{Pi^{1,3}}$ signal will move to a **C, Y, W** states.

9. In a **b/m** state, the controller receiving **$A^{1,3}$** signal will move to a **M** state.

10. In a **b/m** state, the controller receiving **Mco** signal will move to a **c/m** state.

11. In a **b/m** state, the controller receiving **Myp** signal will move to a **y/m** state.

12. In a **b/m** state, the controller receiving **Mixc** signal will move to a **M** state.

13. In a **b/m** state, the controller receiving **Mix** signal will move to a **M** state.

14. In a **b/m** state, the controller receiving **Mix12** signal will move to a **F** state.

15. In a **b/m** state, the controller receiving **Ss** signal will move to a **D** state.

and

16. In a **P** state, the controller receiving **H** signal will move to a **P** state.

17. In a **P** state, the controller receiving **Mu** signal will move to a **P** state.

18. In a **P** state, the controller receiving **Md** signal will move to a **P** state.

19. In a **P** state, the controller receiving **Hi** signal will move to a **M** state.

20. In a **P** state, the controller receiving **Po** signal will move to a **P** state.

21. In a **P** state, the controller receiving **Tu²** signal will move to a **g/w** state.

22. In a **P** state, the controller receiving **S** signal will move to a **M** state.

23. In a **P** state, the controller receiving **Pi**^{1,3} signal will move to a **M** state.

24. In a **P** state, the controller receiving **A**^{1,3} signal will move to a **M** state.

25. In a **P** state, the controller receiving **Mco** signal will move to a **c/m** state.

26. In a **P** state, the controller receiving **Myp** signal will move to a c/**m** state.

27. In a **P** state, the controller receiving **Mixc** signal will move to a **M** state.

28. In a **P** state, the controller receiving **Mix** signal will move to a **M** state.

29. In a **P** state, the controller receiving **Mix12** signal will move to a **F** state.

30. In a **P** state, the controller receiving **Ss** signal will move to a **D** state.

and

31. In a **M** state, the controller receiving **H** signal will not move to a **any** state.

32. In a **M** state, the controller receiving **Mu** signal will not move to a **any** state.

33. In a **M** state, the controller receiving **Md** signal will not move to a **any** state.

34. In a **M** state, the controller receiving **Hi** signal will move to a **M** state.

35. In a **M** state, the controller receiving **Po** signal will move to a **P** state.

36. In a **M** state, the controller receiving **Tu**^{2} signal will move to a **g/w** state.

37. In a **M** state, the controller receiving **S** signal will move to a **M** state.

38. In a **M** state, the controller receiving **Pi**^{1,3} signal will move to a **C, Y, W** states.

39. In a **M** state, the controller receiving **A**^{1,3} signal will move to a **M** state.

40. In a **M** state, the controller receiving **Mco** signal will move to a **c/m** state.

41. In a **M** state, the controller receiving **Myp** signal will move to a c/**m** state.

42. In a **M** state, the controller receiving **Mixc** signal will move to a **M** state.

43. In a **M** state, the controller receiving **Mix** signal will move to a **M** state.

44. In a **M** state, the controller receiving **Mix12** signal will move to a **F** state.

45. In a **M** state, the controller receiving **Ss** signal will move to a **D** state.

and

46. In a **g/w** state, the controller receiving **H** signal will move to a **E** state- empty transition state.

47. In a **g/w** state, the controller receiving **Mu** signal will not move to a **any** state.

48. In a **g/w** state, the controller receiving **Md** signal will not move to a **any** state.

49. In a **g/w** state, the controller receiving **Hi** signal will not move to a **any** state.

50. In a **g/w** state, the controller receiving **Po** signal will not move to a **any** state.

51. In a **g/w** state, the controller receiving $\mathbf{Tu}^{\mathbf{2}}$ signal will move to a **g/w** state.

52. In a **g/w** state, the controller receiving **S** signal will move to a **M** state.

53. In a **g/w** state, the controller receiving **Pi**^{1,3} signal will move to a **C, Y, W** states.

54. In a **g/w** state, the controller receiving **A**^{1,3} signal will move to a **M** state.

55. In a **g/w** state, the controller receiving **Mco** signal will move to a **c/m** state.

56. In a **g/w** state, the controller receiving **Myp** signal will move to a c/**m** state.

57. In a **g/w** state, the controller receiving **Mixc** signal will move to a **M** state.

58. In a **g/w** state, the controller receiving **Mix** signal will move to a **M** state.

59. In a **g/w** state, the controller receiving **Mix12** signal will move to a **F** state.

60. In a **g/w** state, the controller receiving **Ss** signal will move to a **D** state.

and

61. In a **W** state, the controller receiving **H** signal will not move to a **any** state.

62. In a **W** state, the controller receiving **Mu** signal will not move to a **any** state.

63. In a **W** state, the controller receiving **Md** signal will not move to a **any** state.

64. In a **W** state, the controller receiving **Hi** signal will not move to a **any** state.

65. In a **W** state, the controller receiving **Po** signal will not move to a **any** state.

66. In a **W** state, the controller receiving **Tu**^{2} signal will move to a **any** state.

67. In a **W** state, the controller receiving **S** signal will not move to a **any** state.

68. In a **W** state, the controller receiving **Pi**^{1,3} signal will move to a **W** states.

69. In a **W** state, the controller receiving **A**^{1,3} signal will move to a **M** state.

70. In a **W** state, the controller receiving **Mco** signal will move to a **c/m** state.

71. In a **W** state, the controller receiving **Myp** signal will move to a c/**m** state.

72. In a **W** state, the controller receiving **Mixc** signal will move to a **M** state.

73. In a **M** state, the controller receiving **Mix** signal will move to a **M** state.

74. In a **M** state, the controller receiving **Mix12** signal will move to a **F** state.

75. In a **M** state, the controller receiving **Ss** signal will move to a **D** state.

and

76. In a **C** state, the controller receiving **H** signal will not move to a **any** state.

77. In a **C** state, the controller receiving **Mu** signal will not move to a **any** state.

78. In a **C** state, the controller receiving **Md** signal will not move to a **any** state.

79. In a **C** state, the controller receiving **Hi** signal will not move to a **any** state.

80. In a **C** state, the controller receiving **Po** signal will move to a **E** state.

81. In a **C** state, the controller receiving $\mathbf{Tu}^{\mathbf{2}}$ signal will move to an **E** state.

82. In a **C** state, the controller receiving **S** signal will move to an **E** state.

83. In a **C** state, the controller receiving $\mathbf{Pi}^{\mathbf{1,3}}$ signal will move to a **C** states.

84. In a **C** state, the controller receiving $\mathbf{A}^{\mathbf{1,3}}$ signal will move to a **M** state.

85. In a **C** state, the controller receiving **Mco** signal will move to a **c/m** state.

86. In a **C** state, the controller receiving **Myp** signal will move to a c/**m** state.

87. In a **C** state, the controller receiving **Mixc** signal will move to a **M** state.

88. In a **C** state, the controller receiving **Mix** signal will move to a **M** state.

89. In a **C** state, the controller receiving **Mix12** signal will move to a **F** state.

90. In a **C** state, the controller receiving **Ss** signal will move to a **D** state.

and

91. In a **Y** state, the controller receiving **H** signal will not move to a **any** state.

92. In a **Y** state, the controller receiving **Mu** signal will not move to a **any** state.

93. In a **Y** state, the controller receiving **Md** signal will not move to a **any** state.

94. In a **Y** state, the controller receiving **Hi** signal will move to an **E** state.

95. In a **Y** state, the controller receiving **Po** signal will move to an **E** state.

96. In a **Y** state, the controller receiving $\mathbf{Tu^2}$ signal will move to an **E** state.

97. In a **Y** state, the controller receiving **S** signal will move to an **E** state.

98. In a **Y** state, the controller receiving **Pi**^{1,3} signal will move to a **Y** states.

99. In a **Y** state, the controller receiving **A**^{1,3} signal will move to a **M** state.

100. In a **Y** state, the controller receiving **Mco** signal will move to a **c/m** state.

101. In a **Y** state, the controller receiving **Myp** signal will move to a **c/m** state.

102. In a **Y** state, the controller receiving **Mixc** signal will move to a **M** state.

103. In a **Y** state, the controller receiving **Mix** signal will move to a **M** state.

104. In a **Y** state, the controller receiving **Mix12** signal will move to a **F** state.

105.In a **Y** state, the controller receiving **Ss** signal will move to a **D** state.

and

106.In a **c/m** state, the controller receiving **H** signal will not move to a **any** state.

107.In a **c/m** state, the controller receiving **Mu** signal will not move to a **any** state.

108.In a **c/m** state, the controller receiving **Md** signal will not move to a **any** state.

109.In a **c/m** state, the controller receiving **Hi** signal will move to a **E** state.

110.In a **c/m** state, the controller receiving **Po** signal will move to a **E** state.

111. In a **c/m** state, the controller receiving **Tu**^{2} signal will move to an **E** state.

112. In a **c/m** state, the controller receiving **S** signal will move to an **E** state.

113. In a **c/m** state, the controller receiving **Pi**^{1,3} signal will move to an **E** states.

114. In a **c/m** state, the controller receiving **A**^{1,3} signal will move to a **M** state.

115. In a **c/m** state, the controller receiving **Mco** signal will move to a **c/m** state.

116. In a **c/m** state, the controller receiving **Myp** signal will move to a **y/m** state.

117. In a **c/m** state, the controller receiving **Mixc** signal will move to a **M** state.

118. In a **c/m** state, the controller receiving **Mix** signal will move to a **M** state.

119. In a **c/m** state, the controller receiving **Mix12** signal will move to a **F** state.

120. In a **c/m** state, the controller receiving **Ss** signal will move to a **D** state.

and

121. In a **y/m** state, the controller receiving **H** signal will not move to a **any** state.

122. In a **y/m** state, the controller receiving **Mu** signal will not move to a **any** state.

123. In a **y/m** state, the controller receiving **Md** signal will not move to a **any** state.

124. In a **y/m** state, the controller receiving **Hi** signal will move to an **E** state.

125. In a **y/m** state, the controller receiving **Po** signal will move to an **E** state.

126. In a **y/m** state, the controller receiving $\mathbf{Tu}^{\mathbf{2}}$ signal will move to an **E** state.

127. In a **y/m** state, the controller receiving **S** signal will move to an **E** state.

128. In a **y/m** state, the controller receiving $\mathbf{Pi}^{\mathbf{1,3}}$ signal will move to an **E** states.

129. In a **y/m** state, the controller receiving $\mathbf{A}^{\mathbf{1,3}}$ signal will move to an **E** state.

130. In a **y/m** state, the controller receiving **Mco** signal will move to an **E** state.

131.In a **y/m** state, the controller receiving **Myp** signal will move to a **M** state.

132.In a **y/m** state, the controller receiving **Mixc** signal will move to a **M** state.

133.In a **y/m** state, the controller receiving **Mix** signal will move to a **M** state.

134.In a **y/m** state, the controller receiving **Mix12** signal will move to a **F** state.

135.In a **y/m** state, the controller receiving **Ss** signal will move to a **D** state.

and

136.In a **F** state, the controller receiving **H** signal will not move to a **any** state.

137. In a **F** state, the controller receiving **Mu** signal will not move to a **any** state.

138. In a **F** state, the controller receiving **Md** signal will not move to a **any** state.

139. In a **F** state, the controller receiving **Hi** signal will move to an **E** state.

140. In a **F** state, the controller receiving **Po** signal will move to an **E** state.

141. In a **F** state, the controller receiving $\mathbf{Tu}^{\mathbf{2}}$ signal will move to an **E** state.

142. In a **F** state, the controller receiving **S** signal will move to an **E** state.

143. In a **F** state, the controller receiving $\mathbf{Pi}^{\mathbf{1,3}}$ signal will move to an **E** states.

144. In a **F** state, the controller receiving **A**^{1,3} signal will move to an **E** state.

145. In a **F** state, the controller receiving **Mco** signal will move to an **E** state.

146. In a **F** state, the controller receiving **Myp** signal will move to an **E** state.

147. In a **F** state, the controller receiving **Mixc** signal will move to an **E** state.

148. In a **F** state, the controller receiving **Mix** signal will move to an **E** state.

149. In a **F** state, the controller receiving **Mix12** signal will move to a **F** state.

150. In a **F** state, the controller receiving **Ss** signal will move to a **D** state.

and

151. In a **D** state, the controller receiving **H** signal will not move to a **any** state.

152. In a **D** state, the controller receiving **Mu** signal will not move to a **any** state.

153. In a **D** state, the controller receiving **Md** signal will not move to a **any** state.

154. In a **D** state, the controller receiving **Hi** signal will move to an **E** state.

155. In a **D** state, the controller receiving **Po** signal will move to an **E** state.

156. In a **D** state, the controller receiving **Tu²** signal will move to an **E** state.

157. In a **D** state, the controller receiving **S** signal will move to an **E** state.

158. In a **D** state, the controller receiving **Pi**^{1,3} signal will move to an **E** states.

159. In a **D** state, the controller receiving **A**^{1,3} signal will move to an **E** state.

160. In a **D** state, the controller receiving **Mco** signal will move to an **E** state.

161. In a **D** state, the controller receiving **Myp** signal will move to an **E** state.

162. In a **D** state, the controller receiving **Mixc** signal will move to an **E** state.

163. In a **D** state, the controller receiving **Mix** signal will move to an **E** state.

164. In a **D** state, the controller receiving **Mix12** signal will move to an **E** state.

165. In a **D** state, the controller receiving **Ss** signal will move to a **D** state.

5 MACHINE PROCESSING OF WUSAIR AUTOMATON.

Illustration 7 is a state transition diagram of the Pounding State Machine(***PSM***) called M_2. It has eleven states labeled *boy/man, pistol, girl/woman, mortar, fufu, cassava, cassava mix, cocoyam/plantain, cocoyam/plantain mix, water and dish* ,and fifteen conditions labeled as *handle, move-up, move-down, hits, pounds, stirs, turns, picks, adds, mix-cassava-only, mix-cocoplant-only, mix-yp, mix-c, mix,*

mix-1-2 and soften-smooth. The start state of PSM is *boy/man* and it is normally indicated by a pointing arrow to the state *boy/man*. The accept state is the *dish* state and it is normally indicated by double circle around the state. The arrows moving from one state to another is called transitions. When the automaton receives an input string {*boy/man, pistol, girl/woman, mortar, fufu, cassava, cassava mix, cocoyam/plantian, cocoyam/plantain mix, water, dish*}, it processes that string and produces an output. The output is either accept or reject. The processing begins in M_2 start state. The automaton receives the

symbols from the input string one by one from left to right. After ready symbols, M_2 moves from one state to another along the transition that has symbol as its label. When the last symbol now ready it is in the accept state. The processing of M_2 as follows:

1. start in state *boy*/*man*;
2. ready *boy*/*man*, follow transition from *boy*/*man* to *pistol*;
3. ready *pistol*, follow transition from *pistol* to *mortar*;
4. ready *mortar*, follow transition from *mortar* to *cocoplant-mix*;

5. ready *cocoplant-mix*, follow transition from *cocoplant-mix* to *mortar*;
6. ready *mortar*, follow transition from *mortar* to *cassava-mix;*
7. ready *cassava-mix*, follow transition from *cassava-mix* to *mortar*;
8. ready *mortar*, follow transition from *mortar* to *mortar;*
9. ready *mortar*, follow transition from *mortar* to *girl/woman;*
10. ready *girl/woman*, follow transition from *girl/woman* to *cocoplant;*
11. ready *cocoplant*, follow transition from *cocoplant* to *mortar;*

12. ready *mortar*, follow transition from *mortar* to *girl/woman;*
13. ready *girl/woman*, follow transition *girl/woman* to *cassava;*
14. ready *cassava*, follow transition *cassava* to *mortar*;
15. ready *mortar*, follow transition *mortar* to *girl/woman*;
16. ready *girl/woman*, follow transition *girl/woman* to *water*;
17. ready *water*, follow transition from *water* to *mortar*;
18. ready *mortar*, follow transition from *mortar* to *fufu*;

19. ready *fufu*, follow transition from *fufu* to *disk;*
20. accept because M_2 is in an accept state *disk* at the end of the input.

Transition function of Pounding State Machine.

The transition function is used to define the rules of moving. The notation of the transition function is δ(state, input)=state.

1. δ(boy/man, handles) =pistol ;
   ---
2. δ(pistol, moves-up) =pistol ;

3. δ(pistol, moves-down) = pistol;
4. δ(pistol, hits) = mortar;

---

5. δ(mortar, stirs) = girl/woman;
6. δ(mortar, stirs) = girl/woman;
7. δ(mortar, stirs)= girl/woman;
8. δ(mortar, mix) = mortar;
9. δ(mortar, mix-cocoplant-only) = cocoyam/plantain-mix;
10. δ(mortar, mix) = mortar;
11. δ(mortar, mix-cassava-only) = cassava-mix;
12. δ(mortar, mixin-1-2) = fufu;

---

13. δ(girl / woman, picks) = cocoyam/ plantain;
14. δ(girl / woman, picks) = cassava;
15. δ(girl/woman, picks) = water;

---

16. δ(cocoyam/plantain, adds) = mortar;
    ---
17. δ(cassava, picks) = mortar;
18. δ(cassava-mix, mix-c) = mortar;
    ---
19. δ(coccoyam-plantain-mix, mix-yp)= mortar;
    ---
20. δ(fufu, soften-smooths)= disk;
21. δ(disk_outer, disk_inner)= **accept**;

We now arrived at the formal definition of state machine.

*A **state machine** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where*

1. *Q is a finite set called the states,*
2. *Σ Is a finite set called the alphabet,*

3. δ: $Q \times \Sigma \rightarrow Q$ *is the transition function,*
4. $q_0 \in Q$ *is the start state.*
5. $F \subseteq Q$ *is the set of accept states.*

A State Machine is defined as :

1. Q: {*boy/man, pistol, girl/woman, mortar, fufu, cassava, cassava-mix, cocoyam/ plantain, cocoyam/plantain mix, water, dish*}
2. Σ: {*handle, move-up, move-down, hits, pounds, stirs, turns, picks, adds, mix-cassava-only, mix-cocoplant-only, mix-yp, mix-c, mix, mix-1-2 and soften-smooth*}
3. δ: {*check transition functions above*}

4. qo is *boy*/*man*

5. F: {*disk_outer, disk_inner*}

SUMMARY

This chapter is about the simulation of pistol-mortar activity. An introductory text on the pistol-mortar simulation was discussed. Here, I modeled the pounding automaton and reviewed the Wusair Task Orientation. The Wusair automaton is described with the states and conditions of the state machine along with a complete state diagram called Wusair State Machine. A pounding controller for the Wusair state machine is created. The state transition table of Wusair State Machine is

tabulated in the appendix. The pounding automaton model is modeled as the controller moves from state to state with the received input signals. A total of 165 pounding controls was modeled in the Wusair State Machine. Finally is the machine processing of Wusair Automaton. An instance of the Wusair State Machine is created called Pounding State Machine, M_2.

REFERENCES

1. *Frank Appiah* (Nsiah). Pounding Task: Atask-oriented design[1] , series 1 of wusair. Kwame Nkrumah University, Journal of Science and Technology (JUST). 2017*
2. *Frank Appiah* (Nsiah). Pounding Task: A task-oriented design[2] , series 2 of wusair. Kwame Nkrumah University, Journal of Science and Technology (JUST). 2017*
3. *Paul A. Alcorn, Technology, Society and Culture. Person Custom Publishing.2002.*
4. *Rudi Volti, Pitzer College. Society and Technological Change, Second Edition. St. Martin's Press, New York.1992.*

Indexes

A

accept states.....67, 129

adds.....92, 121, 128, 129

alphabet.....67, 128

ALU.....28

analogy.....14

architecture.....14, 27, 41

Architecture.....26

automata.....19, 41, 47

automatic.....14

automatic pen.....20, 21, 48, 49, 68

automation.....20, 48

automaton....14, 19, 25, 30, 47, 54, 63, 66, 68, 122

Automaton.....45

automaton diagram.....68

B

ball.....20, 21, 22, 23, 24, 48, 49, 50, 51, 52

boy.....91

boy/man.....88, 91, 121, 122, 123, 126, 129, 130

C

cassava....74, 80, 88, 89, 91, 92, 93, 121, 122, 124, 125, 127, 128, 129

classic.....39

classic computer.....40

cocoyam 80, 88, 89, 91, 92, 93, 121, 122, 127, 128, 129

complex.....20

complexity.....14, 41

computational.....14, 19, 47

Computational.....45

computational model.....19, 68

computational model 68

computer.....26, 31, 54

computers.....19, 47

computing.....14

computing device.....68

condition.....49

conditions.....14, 37, 91

control.....28, 39

controller.14, 19, 20, 21, 23, 24, 25, 47, 48, 49, 51, 52, 53, 55, 57, 68, 91, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120

Controller.....45

controller functions.....68

couple-pounding.....83

current computer.....28

D

decomposition.....39

design.....14, 30, 82

design automaton.....68

device.....14, 19

diagrams.....14

dish.....89, 91, 121, 122, 129

disk.....79, 82, 126, 128, 130

disk_inner.....128, 130

disk_outer.....128, 130

E

Empty.....32

exercise.....83, 85

F

Fill.....22

finite.....14

Finite automata.....19

finite automaton.....19, 47

finite automaton.....66, 128

finite set.....67, 128

five finger.....37

five units.....28

formal definition.....66, 128

fufu 75, 79, 81, 82, 84, 88, 89, 90, 91, 93, 121, 122, 125, 126, 127, 128, 129

function.....65, 67, 68, 126, 129

G

game.....75, 85, 86

Game.....86, 87

game of pounding.....75, 85

girl.....91

girl/woman.....88, 121, 122, 124, 125, 127, 129

H

hand.....27

hand-side.....14

handle.....88, 91, 121, 129

hits.....89, 91, 121, 127, 129

home.....79

hunting agent.....82

I

information.....29, 30, 31, 42

ink.....19

input.....28, 91

input conditions.....91

interactions.....78

interactions.....31

K

key.....14

key automaton.....25

Key Automaton.....33

keyboard.....27, 30, 54

L

limited.....19

M

machine..14, 20, 25, 27, 28, 29, 35, 36, 38, 39, 42, 48, 68, 74, 75, 82, 90

machine processing.....68

machinery.....31

man.....91

material.....74, 88, 91, 92

mathematical.....14, 19

meaning.....91, 92, 93

memory.....19, 27, 28, 30, 33, 42, 47, 54

mix.....88, 89, 91, 92, 93, 121, 122, 123, 124, 127, 128, 129

mix-1-2.....93, 121, 129

mix-c.....93, 121, 128, 129

mix-cassava-only.....92, 93, 121, 127, 129

mix-cocoplant-only.....92, 93, 121, 127, 129

mix-yp.....93, 121, 128, 129

model.....14, 47, 77

monitor.....33

mortar74, 75, 76, 77, 79, 81, 83, 88, 89, 91, 92, 93, 121, 122, 123, 124, 125, 127, 128, 129

move-down.....91, 121, 129

move-up.....91, 121, 129

N

nature.....41

note.....31, 64, 126

O

observation.....75, 85

output.....28, 29, 63, 122

P

paper.....19, 31, 64, 124

pen.....14, 29, 31, 64, 123

pen machine.....20, 48

pen controller.....19, 20, 47, 48, 68

perfection.....42

picks.....88, 89, 92, 93, 121, 127, 128, 129

pistol……74, 75, 76, 77, 79, 81, 83, 88, 89, 91, 121, 122, 123, 126, 127, 129

plantain .80, 88, 89, 91, 92, 93, 121, 122, 127, 128, 129

player.....77

pounder.....83, 86

pounding.....91, 93

pounding controller.....91

pounding machine.....81

Pounding State Machine.....121, 126

pounding task.....76, 79, 82, 83, 85, 87

pounds.....86, 91, 121, 129

pressure of hitting.....81

process.....41

processing.....63, 64, 122, 123

processor.....28, 39, 40

R

ready.....123, 124, 125, 126

ready cassava.....124, 125

ready cocoplant.....124

ready fufu.....126

ready girl/woman.....124, 125

ready mortar.....123, 124, 125

ready pistol.....123

ready water.....125

records.....31

S

screen.....33

script.....20, 22, 24, 25, 48, 50, 52, 53

self-pounding.....83

self-reproducing automata.....41

self-reproducing machine.....25

self-reproduction.....14

signal.....24

simple machine.....35

simpler machinery.....54

Simulations.....78

soften-smooth.....122, 129

state...20, 22, 23, 24, 25, 41, 48, 50, 51, 52, 53, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 68, 89, 90, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,

115, 116, 117, 118, 119, 120, 121, 122, 123, 126, 129

state diagrams.....25

state machine.....128

State Transition.....22, 32, 35, 50, 57, 90

state transition diagram.....48

State Transition Diagram.....90

state transition diagram 20, 48

state transitions.....25

states.....20, 21, 48, 49, 55, 62, 67, 68, 91, 95, 100, 103, 105, 107, 110, 112, 114, 116, 119, 121, 128, 129

stirs.....91, 121, 127, 129

symbols.....63, 122, 123

systems.....78

T

table.....22

task orientation.....85

task orientation.....85

task-oriented.....82

technology.....77

theory.....47

transition 20, 22, 48, 62, 63, 64, 65, 67, 68, 90, 94, 101, 121, 123, 124, 125, 126, 129

transition function.....65, 67, 68, 126, 129

true.....27

turns.....92, 121, 129

U

universal.....27

V

Von Neumann's.....14

W

water.....86, 88, 89, 91, 92, 121, 122, 125, 127, 129

woman.....91

word.....29

work.....77

write.....64, 123

writing automaton.....54

writing automaton.....25

Wusair.....82, 85, 87, 89, 121